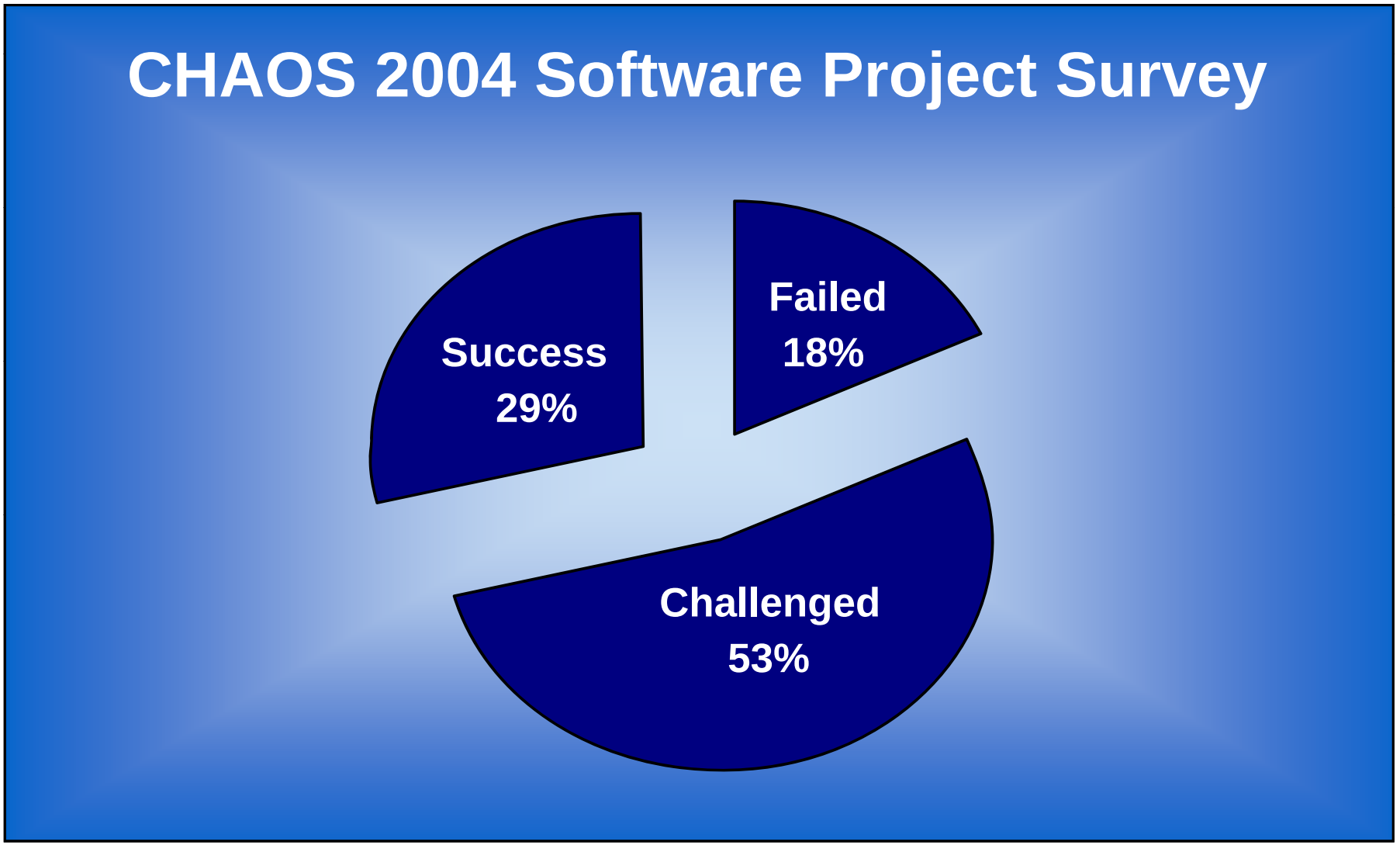
U.S. AIR FORCE

Why Software Projects Fail

Dr. Randall Jensen
Les Dupaix
Software Technology Support Center
Hill AFB, UT

Project success rate



CHAOS Success rate definitions

- **Success**
  - Produced acceptable results delivered close to on-time and on-budget
- **Challenged**
  - Delivered software product significantly over budget and/or schedule
- **Failed**
  - Failed to deliver any usable result within budget or schedule constraints

Software problems are not new

- Unreliable
- Late delivery
- Modification costs prohibitive
- Impossible to maintain
- Inadequate performance
- Product exceeds budget costs

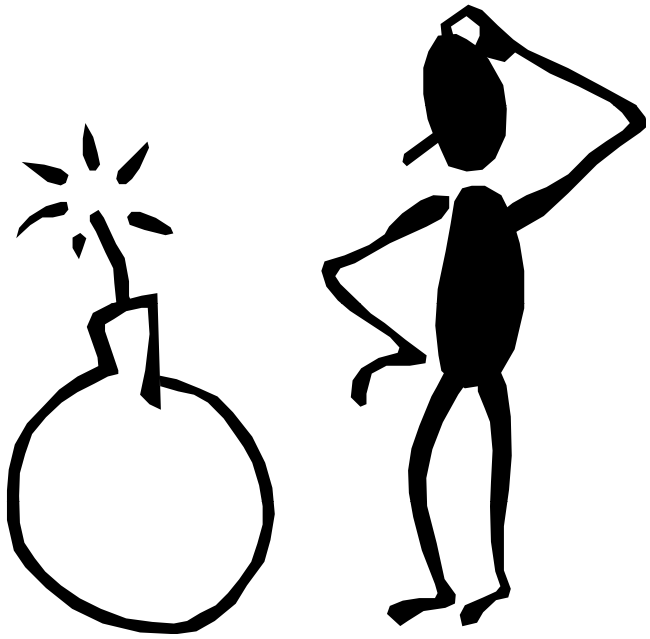
1968 CS Conference, Munich, Germany

Reasons for project failure

- **ADVERTISED**
  - Immature technology
  - Inadequate requirements
  - Insufficient developer experience
- **COVERT**
  - **Poor estimating and planning**
  - **Hope (Pandora's paradox)**

Repeatable things

- Development environment
- Productivity
- Minimum development time
  - Effective product size
  - Complexity
  - Paul Masson rule

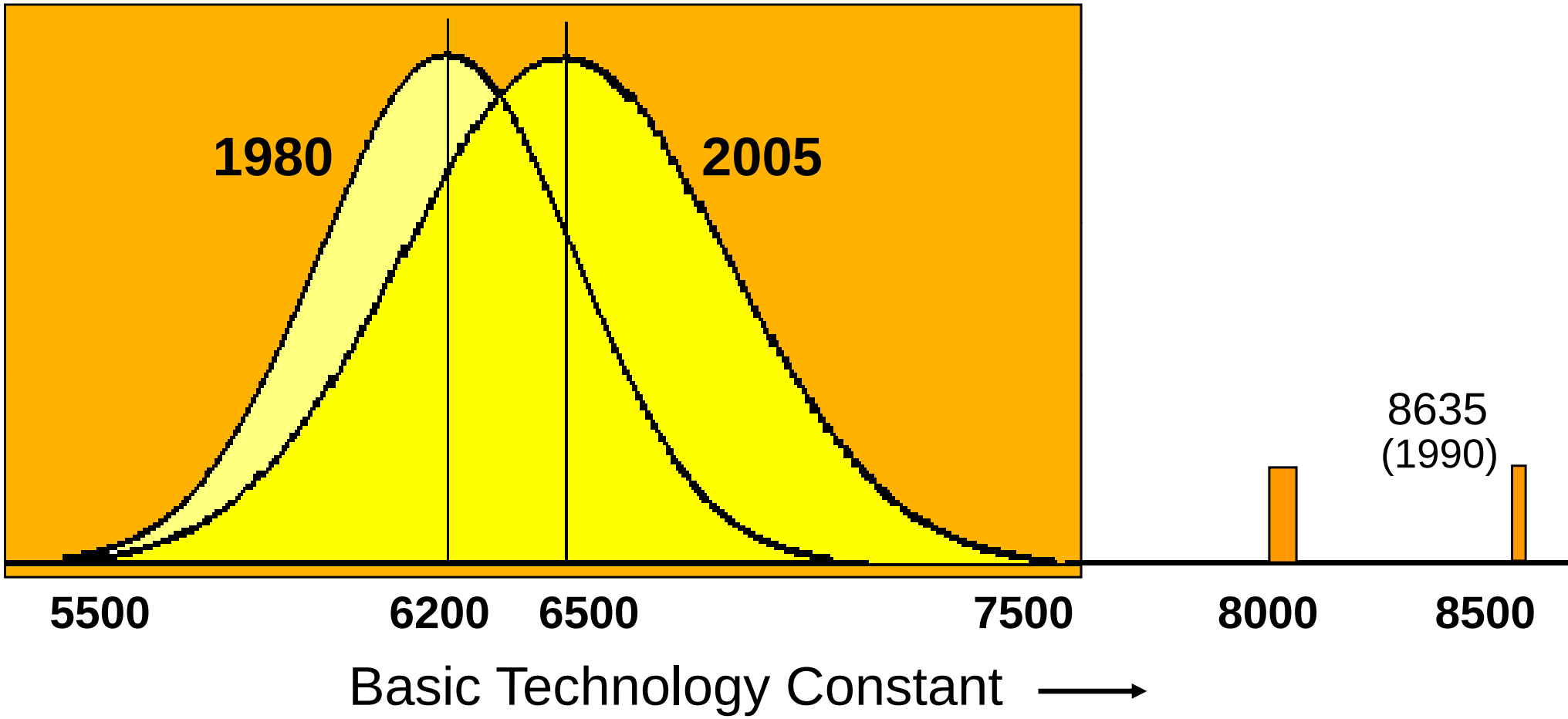


Let’s look at some proof

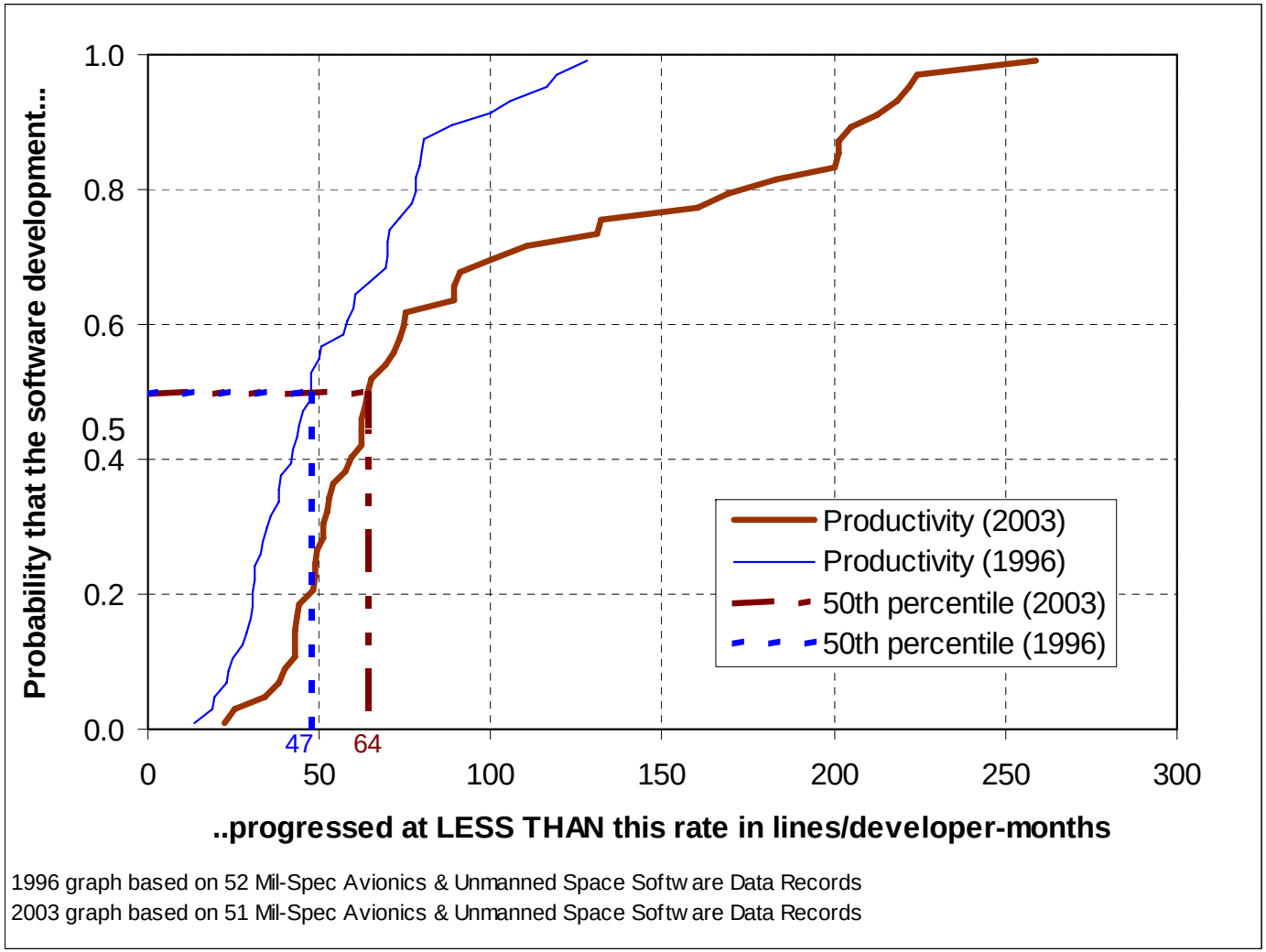
BIG productivity drivers

- Analyst capability
  - Management style
  - Motivation
  - Problem solving skills
  - Use of team methods
  - Working environment
- Application domain experience
- Automated tool support
- Programmer capability
- Use of modern practices

Capability shift (environment)



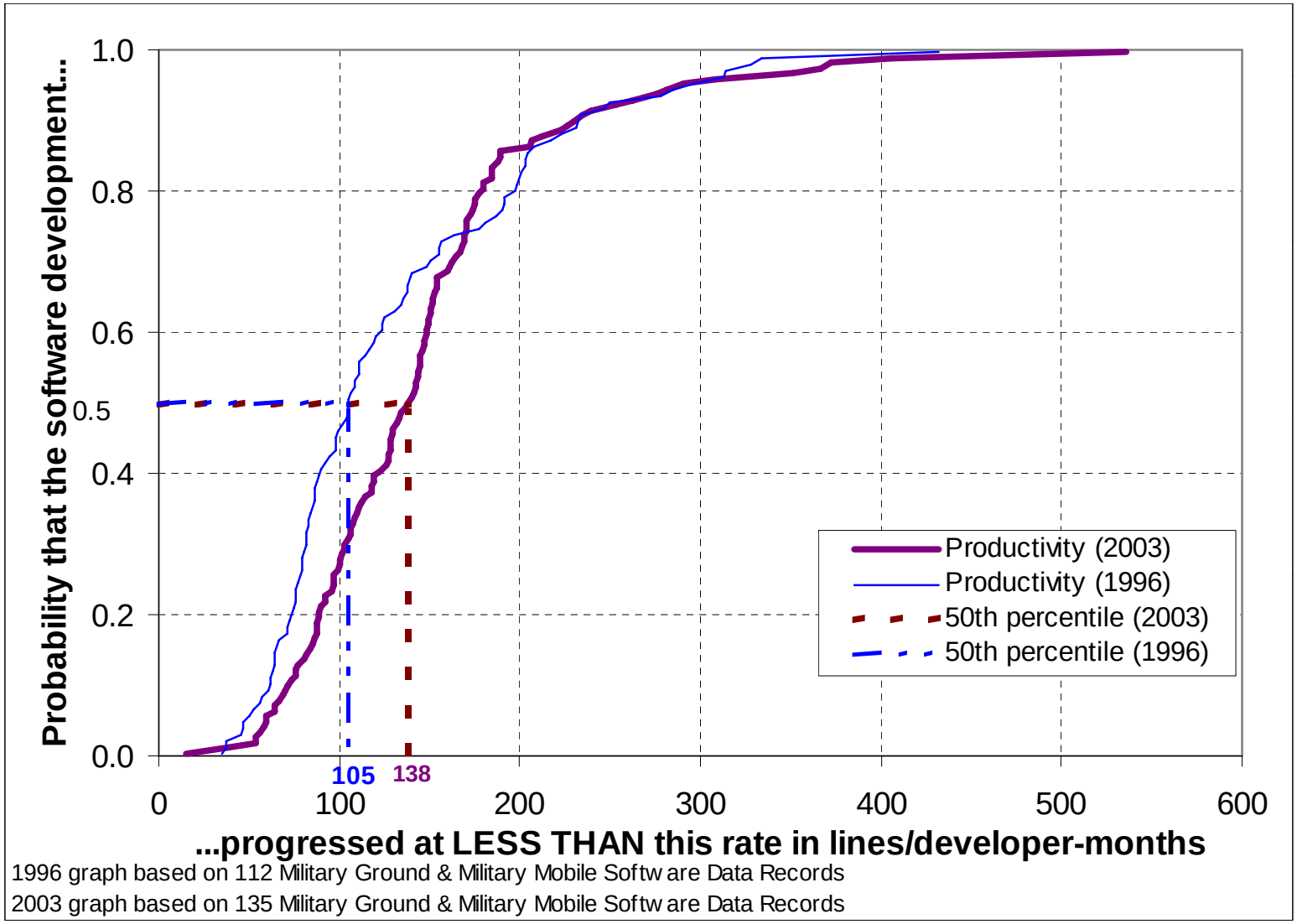
1996/2003 Productivity for Avionics and Unmanned Space



1996 graph based on 52 Mil-Spec Avionics & Unmanned Space Software Data Records
2003 graph based on 51 Mil-Spec Avionics & Unmanned Space Software Data Records

Aerospace Corp – Long etal 2004

1996/2003 Productivity Distributions for Ground and Mobile



1996 graph based on 112 Military Ground & Military Mobile Software Data Records
2003 graph based on 135 Military Ground & Military Mobile Software Data Records

Aerospace, Long et al, 2004

Common technology claim

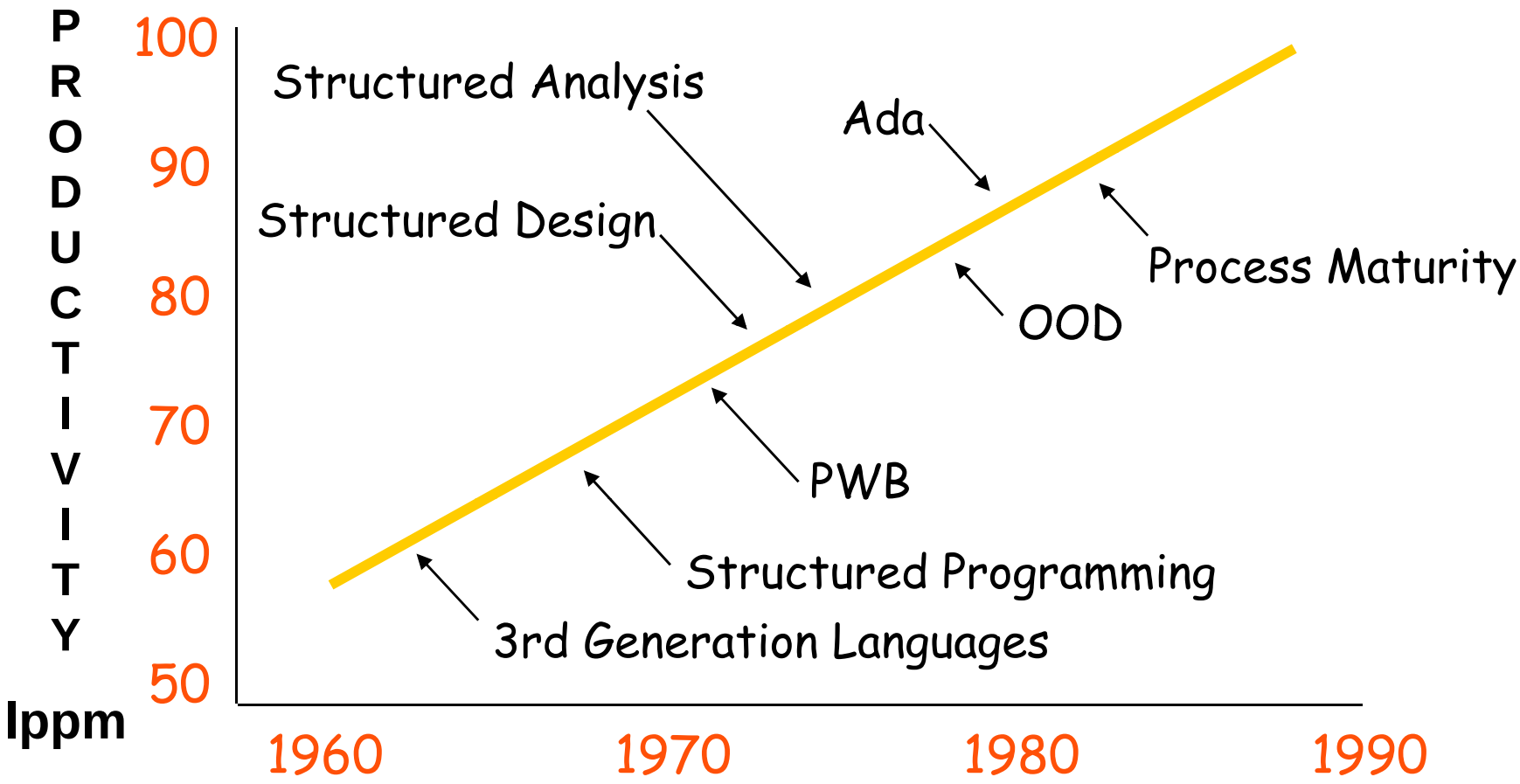
If you use (*new technology*),

Productivity will improve by an *order of magnitude*

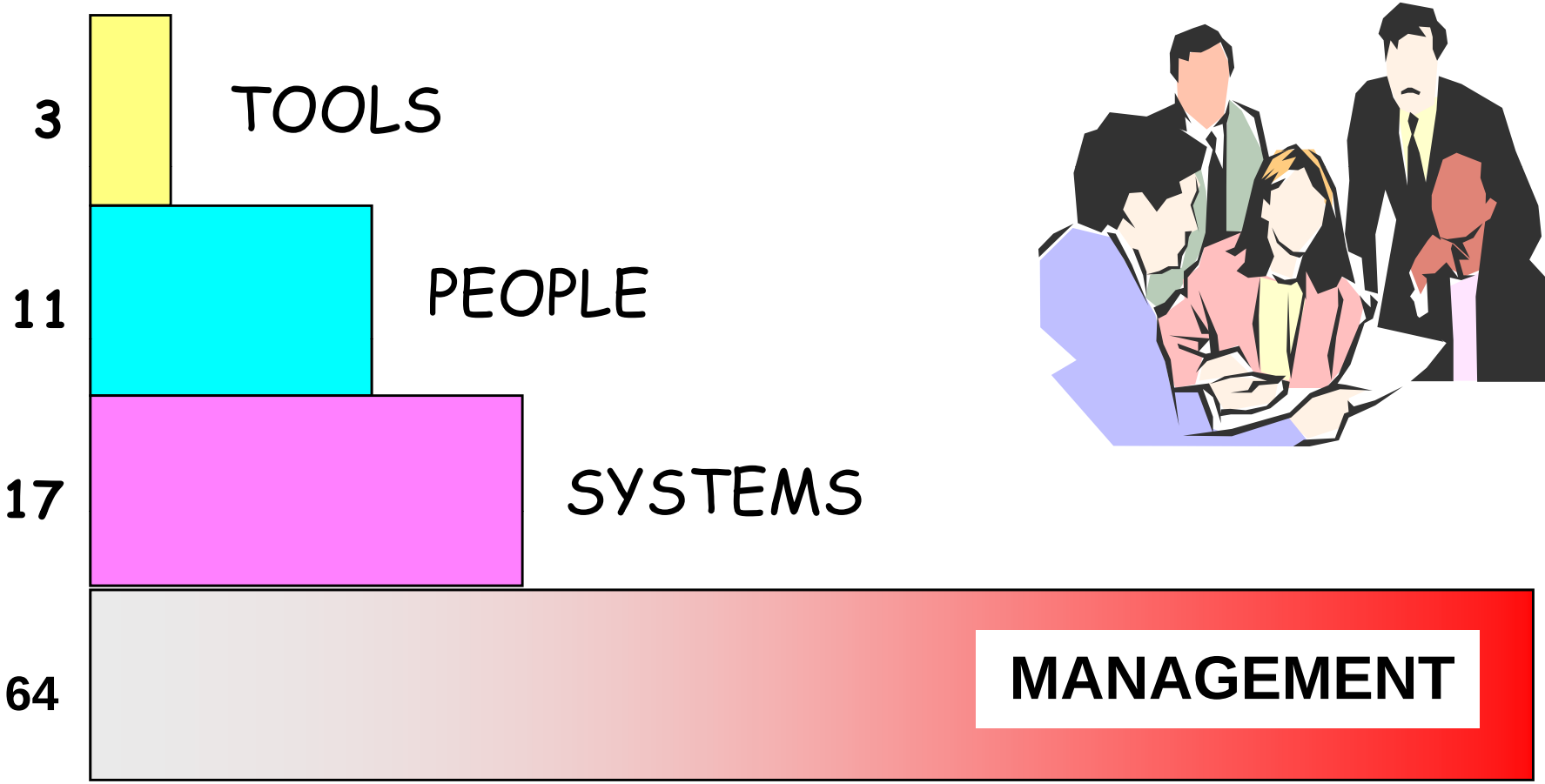
And

Defects will reduce to *zero*

There is always HOPE

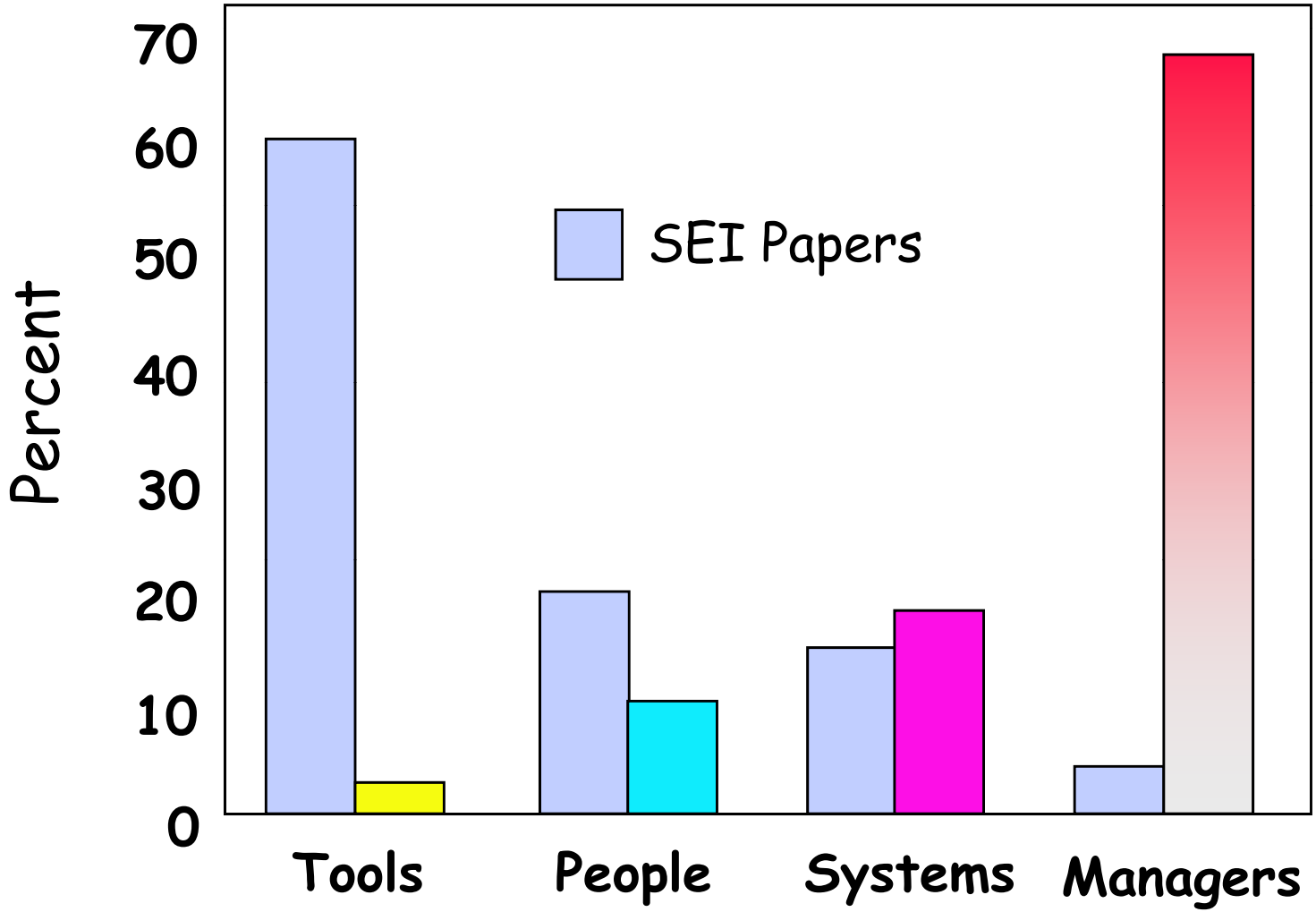


Where would you focus effort?



Source: G. Weinberg, Quality Software Management, Vol. 3

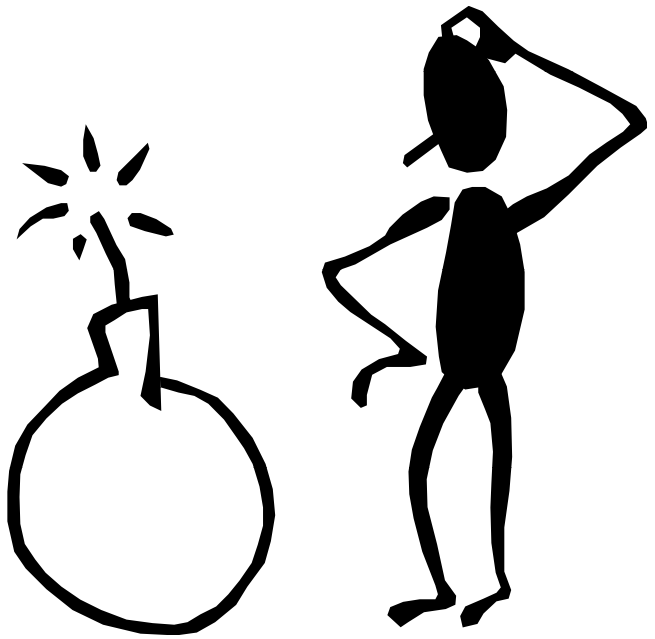
Relative payoff



Source: G. Weinberg, Quality Software Management, Vol. 3

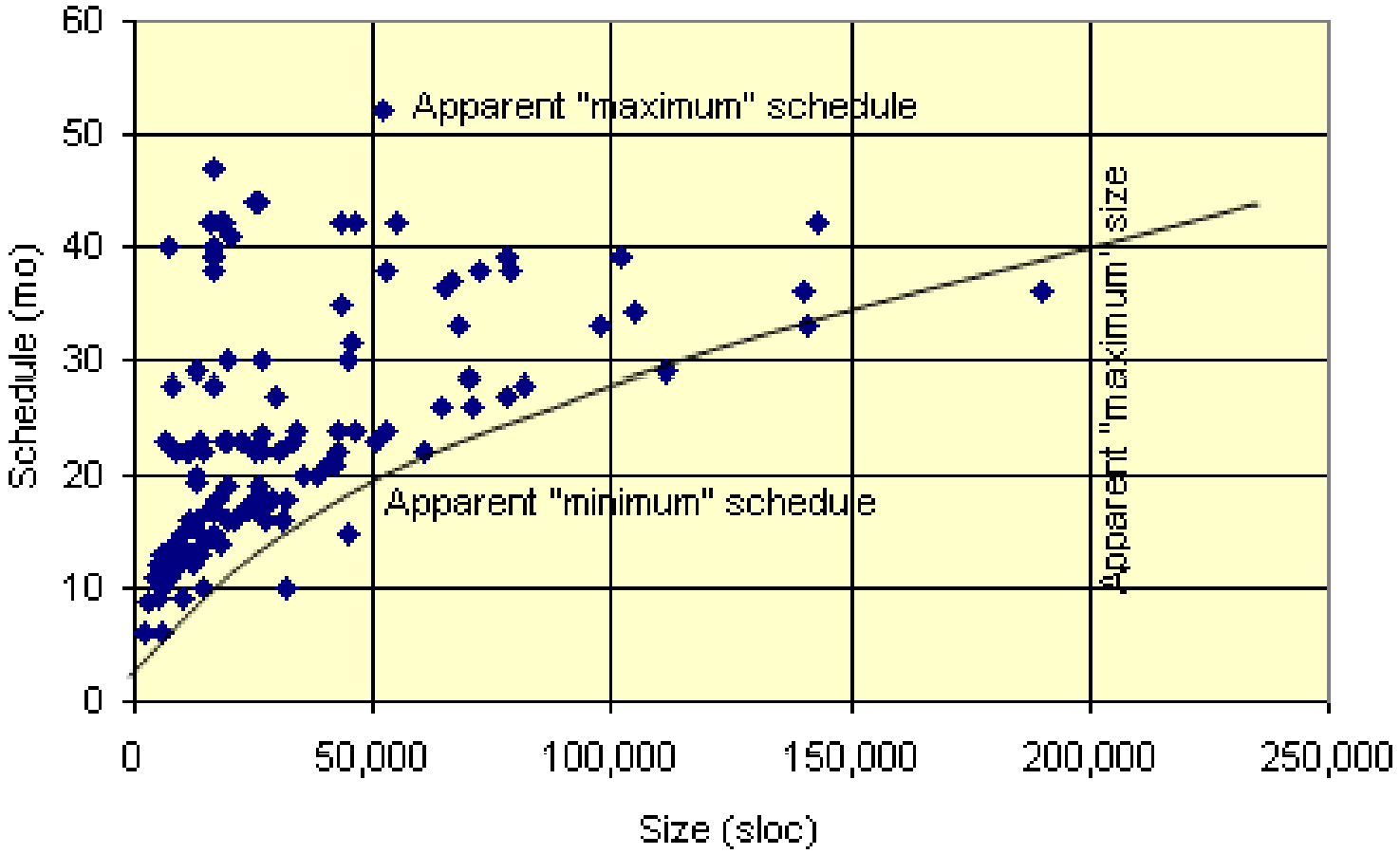
Repeatable things

- Development environment
- Productivity
- Minimum development time
  - Effective product size
  - Complexity
  - Paul Masson rule



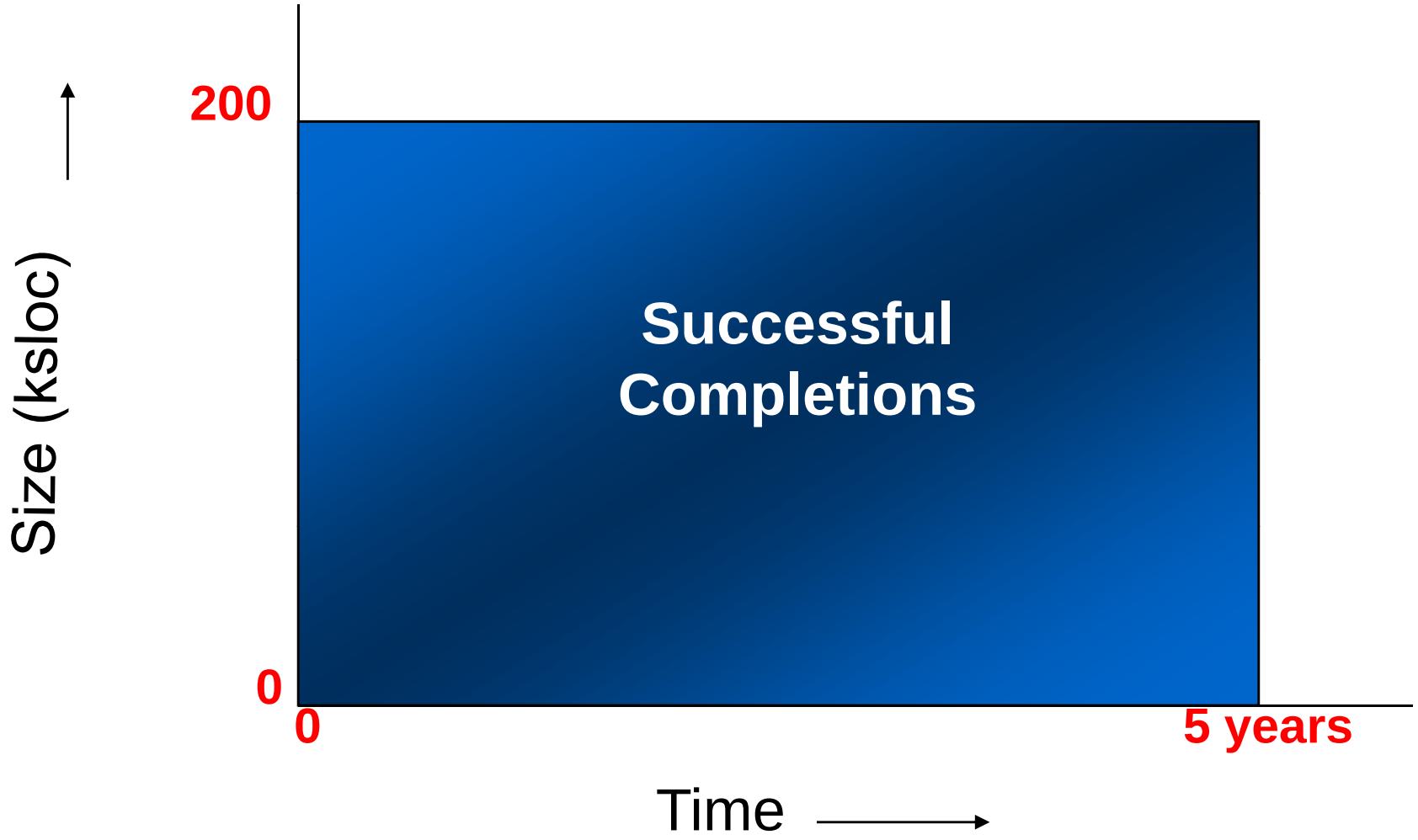
Let’s look at more proof

Historic project data



Source: Long, L. G. et al, 2004

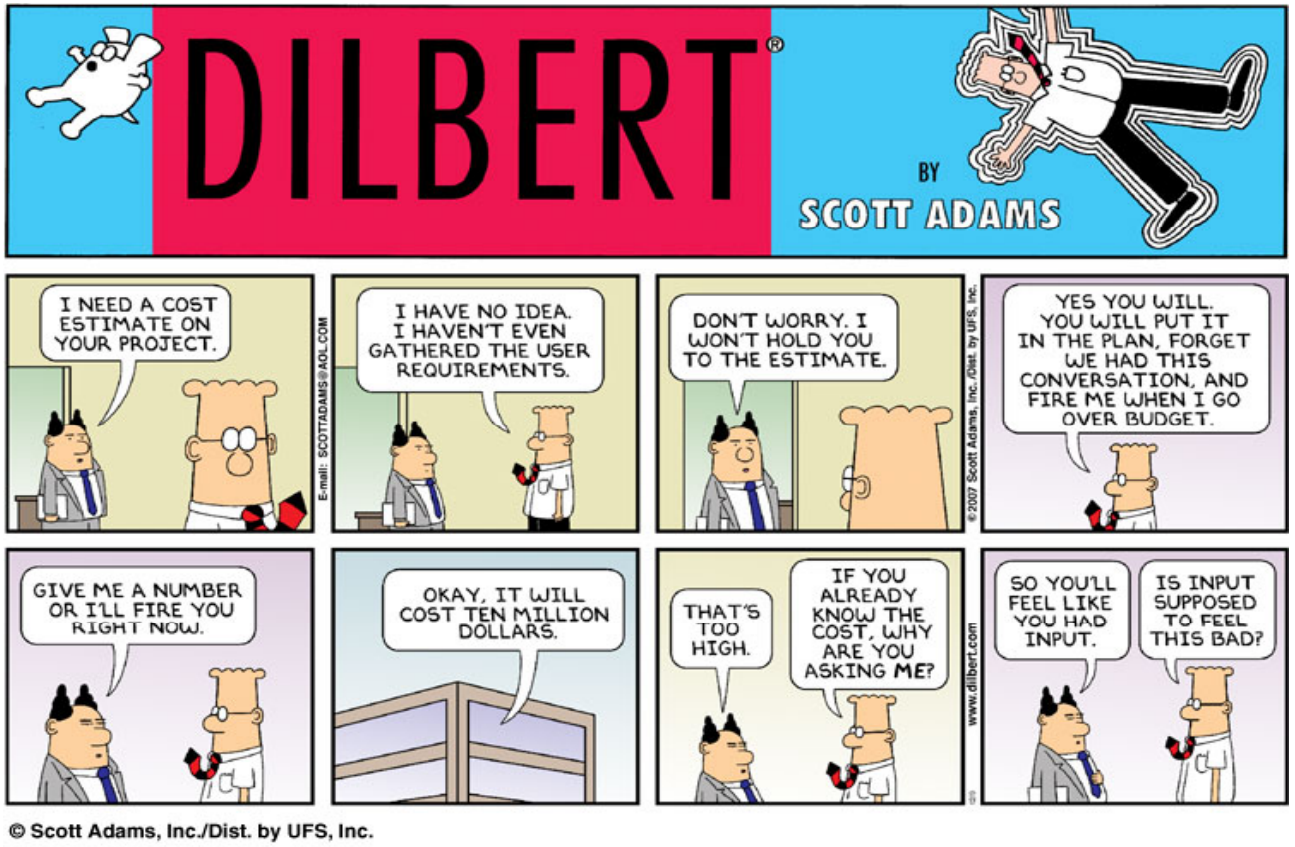
Historic space project limits



Source: Long, L. G. et al, 2004

Three development variables

- Cost
- Schedule
- Scope



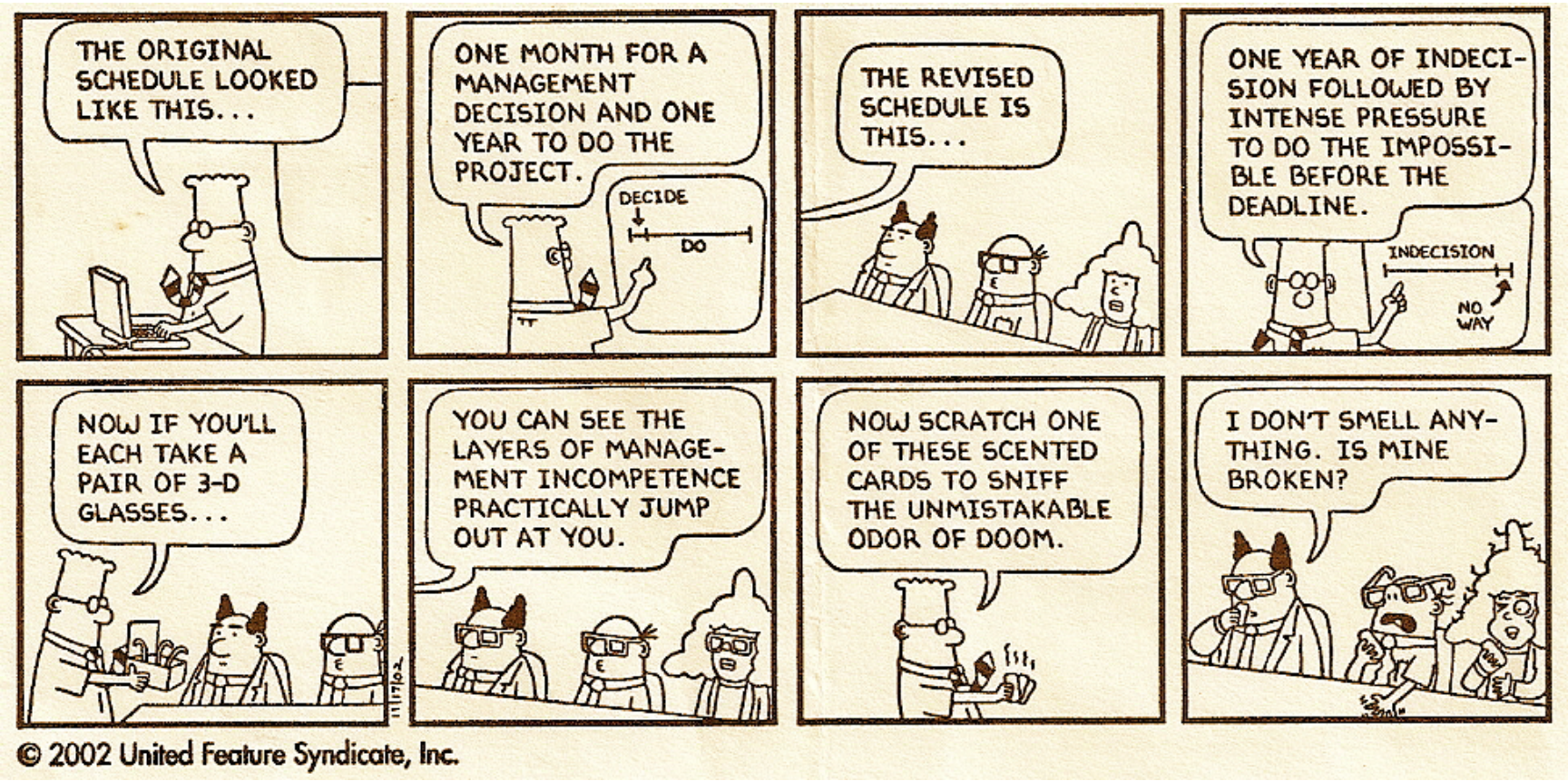
- Developer can control any <u>two</u>

Historic note:

"More software projects have gone awry for lack of calendar time than for all other causes combined…"

F. P. Brooks, Jr., Mythical Man Month

Common estimate dilemma



Constraint analysis

- Success and failure

  determined by

- Expected cost and schedule

  determined by

- Project plan

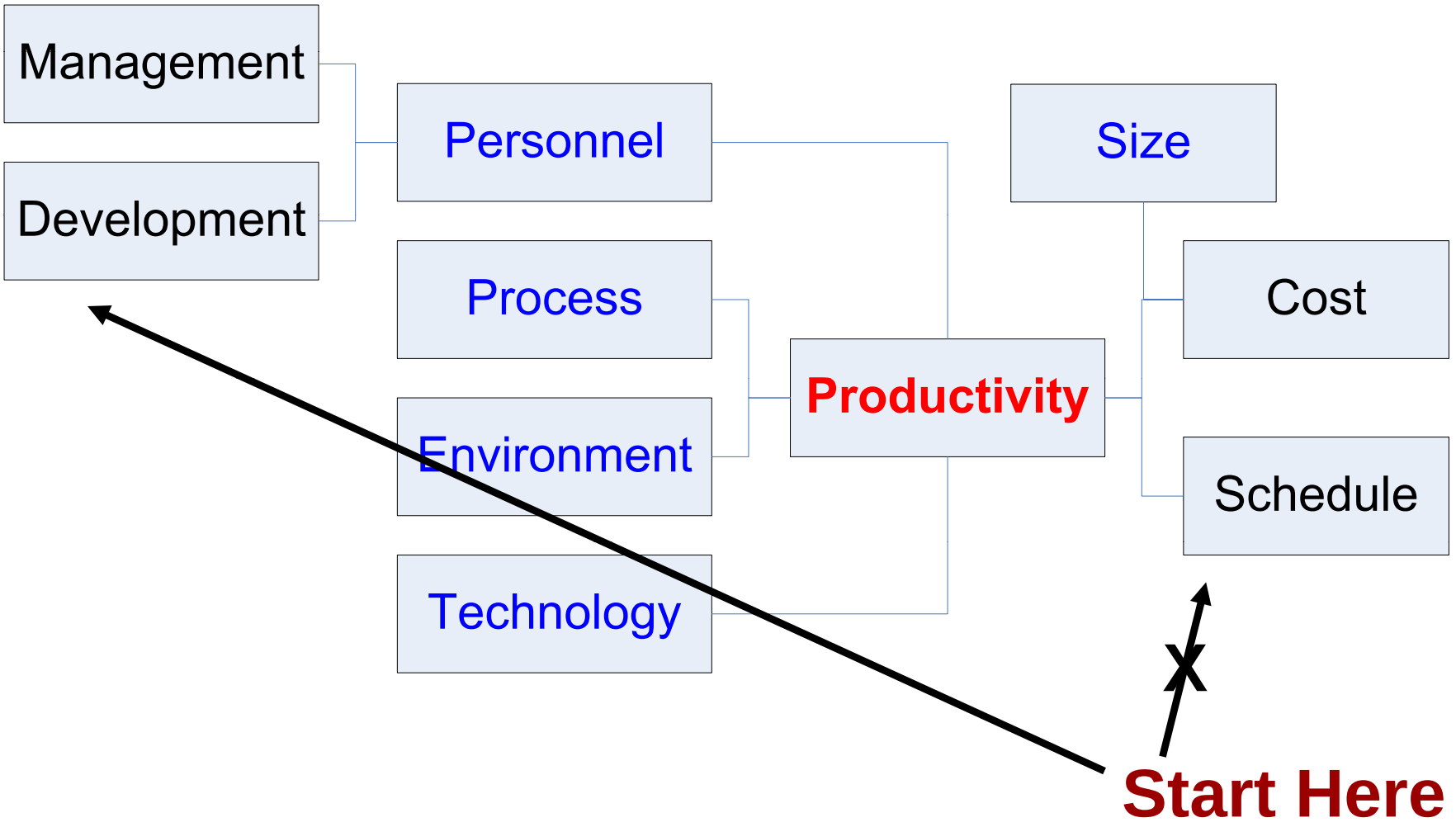
  determined by

- Cost and schedule ESTIMATES

  determined by

- **Managers and/or Estimators**

Key estimate factors



Elements for successful estimates

- Basic understanding of the requirements
- Ability to accurately size the deliverable product
- Assessment of the deliverable complexity
- Profile of the organization's delivery capability

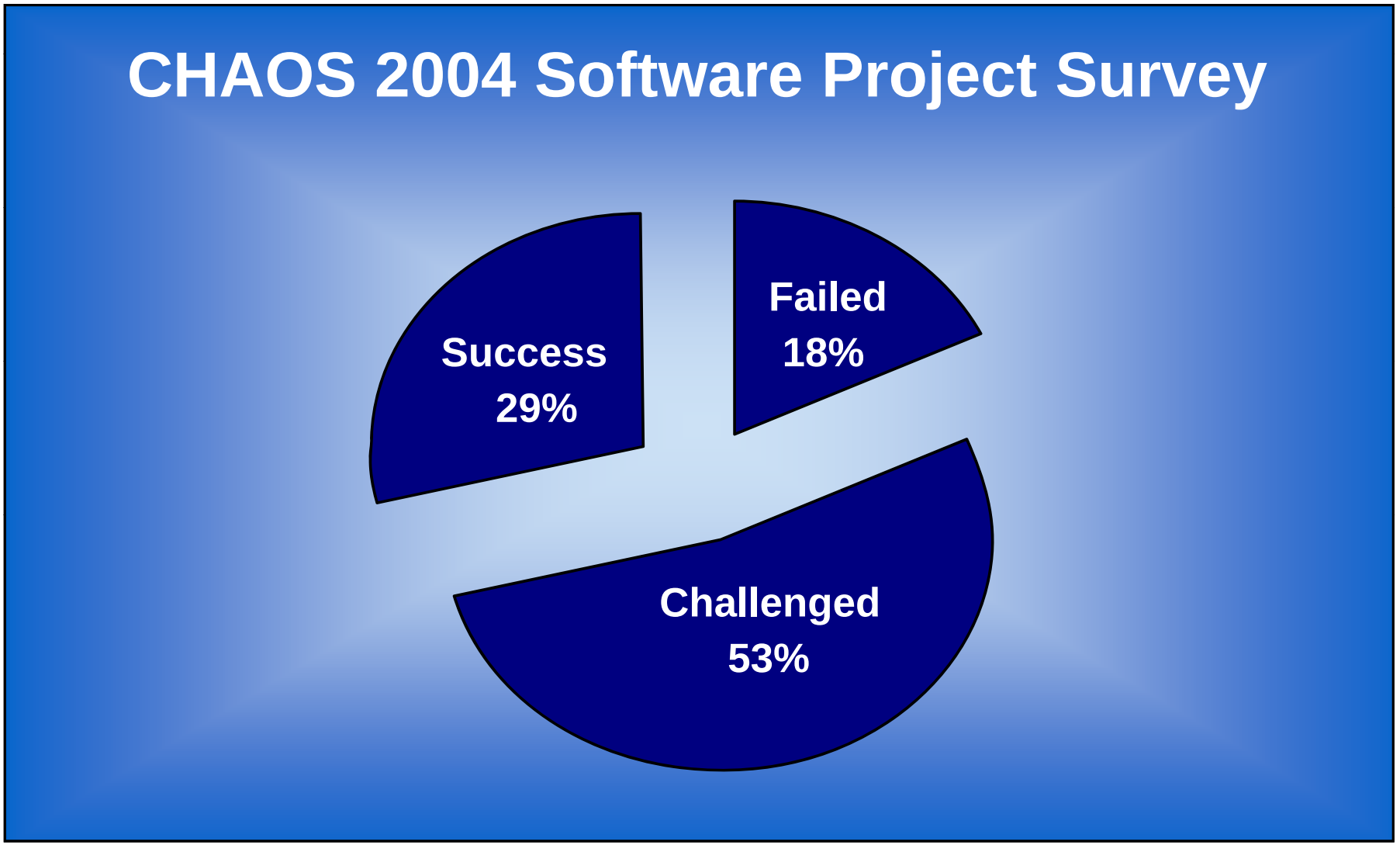
Estimates are important

Without well thought out estimates, there is no firm basis for:

- Defining costs and schedules
- Making tradeoffs
- Associating development costs with the benefits
- Conforming to a budget or schedule

- THE PROJECT IS ALREADY OUT OF CONTROL!

Project success rate



Success definitions: 2nd look

- **Success**
  - **Produced acceptable results delivered close to on-time and on-budget**
- **Challenged**
  - **Delivered software product significantly over budget and/or schedule**
- **Failed**
  - **Failed to deliver any usable result within budget or schedule constraints**

Conclusions

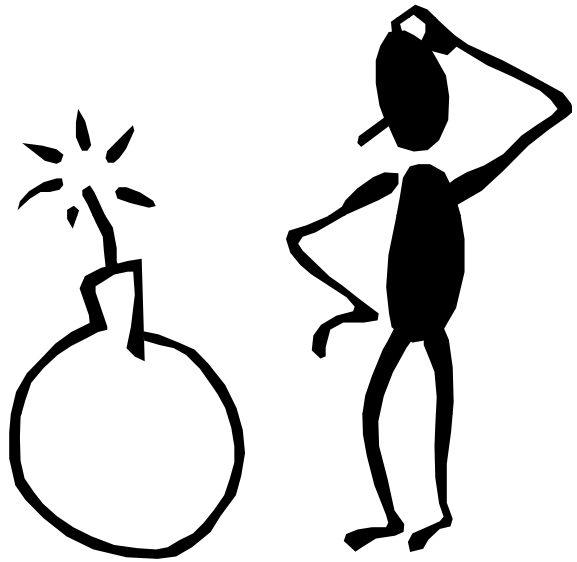
- Software project failures are primarily due to failed expectations
- Technology, requirements and experience are largely accounted for in resource estimates
- Failure ranking:

1. **Poor estimating and planning**
2. **Hope (Not a management strategy)**
3. **Immature technology**
4. **Inadequate requirements**
5. **Insufficient developer capability (Ebonians)**

New estimating models?

- Lack of confidence in existing tools
  - Optimistic estimates
  - Unacceptable estimates
    - Skill
    - Experience
    - Integrity
- Aging tools
  - Quality data
  - Culture is constant
  - New models require validation
- Silver bullet
- New estimators?

Historic note:



We learn from experience that we don't learn from experience.

D. H. Lawrence

The End,

or

is it

The Beginning?

(History does repeat itself)